# Better Language Models with Model Merging

**Thorsten Brants**
Universität des Saarlandes, Computational Linguistics
P.O.Box 151150, D-66041 Saarbrücken, Germany
thorsten@coli.uni-sb.de

*Conference on Empricial Methods in NLP*
*May 17 – 18, 1996, Philadelphia, PA.*

## Abstract

This paper investigates model merging, a technique for deriving Markov models from text or speech corpora. Models are derived by starting with a large and specific model and by successively combining states to build smaller and more general models. We present methods to reduce the time complexity of the algorithm and report on experiments on deriving language models for a speech recognition task. The experiments show the advantage of model merging over the standard bigram approach. The merged model assigns a lower perplexity to the test set and uses considerably fewer states.

## Introduction

Hidden Markov Models are commonly used for statistical language models, e.g. in part-of-speech tagging and speech recognition (Rabiner, 1989). The models need a large set of parameters which are induced from a (text-) corpus. The parameters should be optimal in the sense that the resulting models assign high probabilities to seen training data as well as new data that arises in an application.

There are several methods to estimate model parameters. The first one is to use each word (type) as a state and estimate the transition probabilities between two or three words by using the relative frequencies of a corpus. This method is commonly used in speech recognition and known as word-bigram or word-trigram model. The relative frequencies have to be smoothed to handle the sparse data problem and to avoid zero probabilities.

The second method is a variation of the first method. Words are automatically grouped, e.g. by similarity of distribution in the corpus (Pereira et al., 1993). The relative frequencies of pairs or triples of groups (categories, clusters) are used as model parameters, each group is represented by a state in the model. The second method has the advantage of drastically reducing the number of model parameters and thereby reducing the sparse data problem; there is more data per group than per word, thus estimates are more precise.

The third method uses manually defined categories. They are linguistically motivated and usually called *parts-of-speech*. An important difference to the second method with automatically derived categories is that with the manual definition a word can belong to more than one category. A corpus is (manually) tagged with the categories and transition probabilities between two or three categories are estimated from their relative frequencies. This method is commonly used for part-of-speech tagging (Church, 1988).

The fourth method is a variation of the third method and is also used for part-of-speech tagging. This method does not need a pre-annotated corpus for parameter estimation. Instead it uses a lexicon stating the possible parts-of-speech for each word, a raw text corpus, and an initial bias for the transition and output probabilities. The parameters are estimated by using the Baum-Welch algorithm (Baum et al., 1970). The accuracy of the derived model depends heavily on the initial bias, but with a good choice results are comparable to those of method three (Cutting et al., 1992).

This paper investigates a fifth method for estimating natural language models, combining the advantages of the methods mentioned above. It is suitable for both speech recognition and part-of-speech tagging, has the advantage of automatically deriving word categories from a corpus and is capable of recognizing the fact that a word belongs to more than one category. Unlike other techniques it not only induces transition and output probabilities, but also the model topology, i.e., the number of states, and for each state the outputs that have a non-zero probability. The method is called model merging and was introduced by (Omohundro, 1992).

The rest of the paper is structured as follows. We first give a short introduction to Markov mo-

dels and present the model merging technique. Then, techniques for reducing the time complexity are presented and we report two experiments using these techniques.

## Markov Models

A discrete output, first order *Markov Model* consists of

- a finite set of states $\mathcal{Q} \cup \{q_s, q_e\}$, $q_s, q_e \notin \mathcal{Q}$, with $q_s$ the start state, and $q_e$ the end state;
- a finite output alphabet $\Sigma$;
- a $(|\mathcal{Q}|+1) \times (|\mathcal{Q}|+1)$ matrix, specifying the probabilities of state transitions $p(q'|q)$ between states $q$ and $q'$ (there are no transitions into $q_s$, and no transitions originating in $q_e$); for each state $q \in \mathcal{Q} \cup \{q_s\}$, the sum of the outgoing transition probabilities is 1, $\sum_{q' \in \mathcal{Q} \cup \{q_e\}} p(q'|q) = 1$;
- a $|\mathcal{Q}| \times |\Sigma|$ matrix, specifying the output probabilities $p(\sigma|q)$ of state $q$ emitting output $\sigma$; for each state $q \in \mathcal{Q}$, the sum of the output probabilities is 1, $\sum_{\sigma \in \Sigma} p(\sigma|q) = 1$.

A Markov model starts running in the start state $q_s$, makes a transition at each time step, and stops when reaching the end state $q_e$. The transition from one state to another is done according to the probabilities specified with the transitions. Each time a state is entered (except the start and end state) one of the outputs is chosen (again according to their probabilities) and emitted.

### Assigning Probabilities to Data

For the rest of the paper, we are interested in the probabilities which are assigned to sequences of outputs by the Markov models. These can be calculated in the following way.

Given a model $M$, a sequence of outputs $\sigma = \sigma_1 \ldots \sigma_k$ and a sequence of states $Q = q_1 \ldots q_k$ (of same length), the probability that the model running through the sequence of states and emitting the given outputs is

$$P_M(Q, \sigma) = \left( \prod_{i=1}^{k} p_M(q_i|q_{i-1}) p_M(\sigma_i|q_i) \right) p_M(q_e|q_i)$$

(with $q_0 = q_s$). A sequence of outputs can be emitted by more than one sequence of states, thus we have to sum over all sequences of states with the given length to get the probability that a model emits a given sequence of outputs:

$$P_M(\sigma) = \sum_{Q} P_M(Q, \sigma).$$

The probabilities are calculated very efficiently with the Viterbi algorithm (Viterbi, 1967). Its time complexity is linear to the sequence length despite the exponential growth of the search space.

### Perplexity

Markov models assign rapidly decreasing probabilities to output sequences of increasing length. To compensate for different lengths and to make their probabilities comparable, one uses the perplexity $PP$ of an output sequence instead of its probability. The perplexity is defined as

$$PP_M(\sigma) = \frac{1}{\sqrt[k]{P_M(\sigma)}}.$$

The probability is normalized by taking the $k^{th}$ root ($k$ is the length of the sequence). Similarly, the log perplexity $LP$ is defined:

$$LP_M(\sigma) = \log PP_M(\sigma) = \frac{-\log P_M(\sigma)}{k}.$$

Here, the log probability is normalized by dividing by the length of the sequence.

$PP$ and $LP$ are defined such that higher perplexities (log perplexities, resp.) correspond to lower probabilities, and vice versa. These measures are used to determine the quality of Markov models. The lower the perplexity (and log perplexity) of a test sequence, the higher its probability, and thus the better it is predicted by the model.

## Model Merging

Model merging is a technique for inducing model parameters for Markov models from a text corpus. It was introduced in (Omohundro, 1992) and (Stolcke and Omohundro, 1994) to induce models for regular languages from a few samples, and adapted to natural language models in (Brants, 1995). Unlike other techniques it not only induces transition and output probabilities from the corpus, but also the model topology, i.e., the number of states and for each state the outputs that have non-zero probability. In $n$-gram approaches the topology is fixed. E.g., in a pos-$n$-gram model, the states are mostly syntactically motivated, each state represents a syntactic category and only words belonging to the same category have a non-zero output probability in a particular state. However the $n$-gram-models make the implicit assumption that all words belonging to the same category have a similar distribution in a corpus. This is not true in most of the cases.

By estimating the topology, model merging groups words into categories, since all words that can be emitted by the same state form a category. The advantage of model merging in this respect

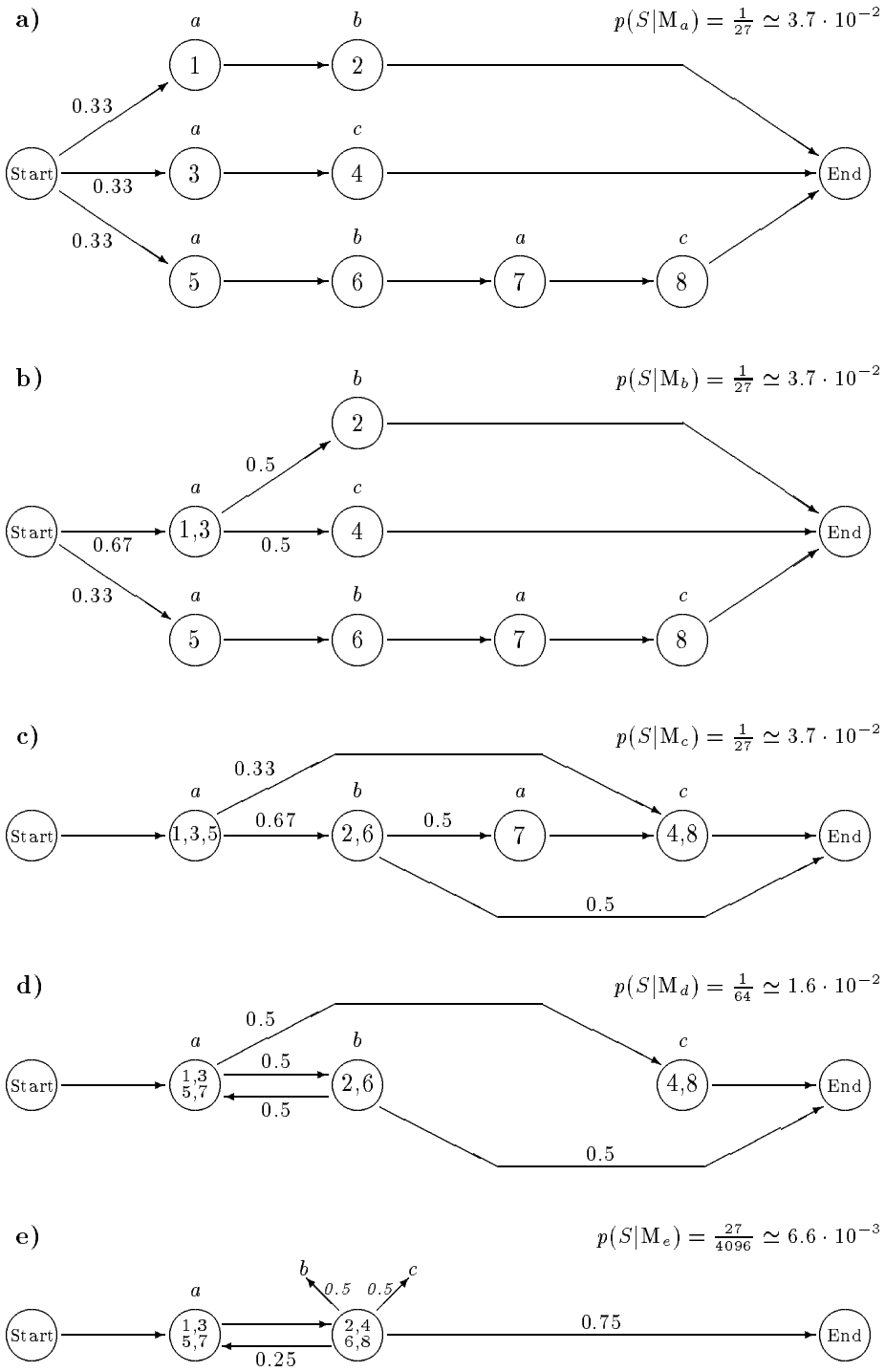


Figure 1: Model merging for a corpus $S = \{ab, ac, abac\}$, starting with the trivial model in a) and ending with the generalization $(a(b|c))^+$ in e). Several steps of merging between model b) and c) are not shown. Unmarked transitions and outputs have probability 1.

is that it can recognize that a word (the type) belongs to more than one category, while each occurrence (the token) is assigned a unique category. This naturally reflects manual syntactic categorizations, where a word can belong to several syntactic classes but each occurrence of a word is unambiguous.

### The Algorithm

Model merging induces Markov models in the following way. Merging starts with an initial, very general model. For this purpose, the maximum likelihood Markov model is chosen, i.e., a model that exactly matches the corpus. There is one path for each utterance in the corpus and each path is used by one utterance only. Each path gets the same probability $1/u$, with $u$ the number of utterances in the corpus. This model is also referred to as the trivial model. Figure 1.a shows the trivial model for a corpus with words $a, b, c$ and utterances $ab, ac, abac$. It has one path for each of the three utterances $ab$, $ac$, and $abac$, and each path gets the same probability $1/3$. The trivial model assigns a probability of $p(S|\mathrm{M}_a) = 1/27$ to the corpus. Since the model makes an implicit independence assumption between the utterances, the corpus probability is calculated by multiplying the utterance's probabilities, yielding $1/3 \cdot 1/3 \cdot 1/3 = 1/27$.

Now states are merged successively, except for the start and end state. Two states are selected and removed and a new merged state is added. The transitions from and to the old states are redirected to the new state, the transition probabilities are adjusted to maximize the likelihood of the corpus; the outputs are joined and their probabilities are also adjusted to maximize the likelihood. One step of merging can be seen in figure 1.b. States 1 and 3 are removed, a combined state 1,3 is added, and the probabilities are adjusted.

The criterion for selecting states to merge is the probability of the Markov model generating the corpus. We want this probability to stay as high as possible. Of all possible merges (generally, there are $k(k-1)/2$ possible merges, with $k$ the number of states exclusive start and end state which are not allowed to merge) we take the merge that results in the minimal change of the probability. For the trivial model and $u$ pairwise different utterances the probability is $p(S|M_{triv}) = 1/u^u$. The probability either stays constant, as in Figure 1.b and c, or decreases, as in 1.d and e. The probability never increases because the trivial model is the maximum likelihood model, i.e., it maximizes the probability of the corpus given the model.

Model merging stops when a predefined threshold for the corpus probability is reached. Some statistically motivated criteria for termination using model priors are discussed in (Stolcke and Omohundro, 1994).

## Using Model Merging

The model merging algorithm needs several optimizations to be applicable to large natural language corpora, otherwise the amount of time needed for deriving the models is too large. Generally, there are $O(l^2)$ hypothetical merges to be tested for each merging step ($l$ is the length of the training corpus). The probability of the training corpus has to be calculated for each hypothetical merge, which is $O(l)$ with dynamic programming. Thus, each step of merging is $O(l^3)$. If we want to reduce the model from size $l + 2$ (the trivial model, which consists of one state for each token plus initial and final states) to some fixed size, we need $O(l)$ steps of merging. Therefore, deriving a Markov model by model merging is $O(l^4)$ in time.

(Stolcke and Omohundro, 1994) discuss several computational shortcuts and approximations:

1. Immediate merging of identical initial and final states of different utterances. These merges do not change the corpus probability and thus are the first merges anyway.
2. Usage of the Viterbi path (best path) only instead of summing up all paths to determine the corpus probability.
3. The assumption that all input samples retain their Viterbi path after merging. Making this approximation, it is no longer necessary to reparse the whole corpus for each hypothetical merge.

We use two additional strategies to reduce the time complexity of the algorithm: a series of cascaded constraints on the merges and the variation of the starting point.

### Constraints

When applying model merging one can observe that first mainly states with the same output are merged. After several steps of merging, it is no longer the same output but still mainly states that output words of the same syntactic category are merged. This behavior can be exploited by introducing constraints on the merging process. The constraints allow only some of the otherwise possible merges. Only the allowed merges are tested for each step of merging.

We consider constraints that divide the states of the current model into equivalence classes. Only states belonging to the same class are allowed to merge. E.g., we can divide the states into classes

generating the same outputs. If the current model has $N$ states and we divide them into $k > 1$ non-empty equivalence classes $C_1 \ldots C_k$, then, instead of $N(N-1)/2$, we have to test

$$\sum_{i=1}^{k} \frac{|C_i|(|C_i|-1)}{2} < \frac{N(N-1)}{2}$$

merges only.

The best case for a model of size $N$ is the division into $N/2$ classes of size 2. Then, only $N/2$ merges must be tested to find the best merge.

The best division into $k > 1$ classes for some model of size $N$ is the creation of classes that all have the same size $N/k$ (or an approximation if $N/k \notin \mathbb{N}$). Then,

$$\frac{\frac{N}{k}(\frac{N}{k}-1)}{2} \cdot k = \frac{N(\frac{N}{k}-1)}{2}$$

must be tested for each step of merging.

Thus, the introduction of these constraints does not reduce the order of the time complexity, but it can reduce the constant factor significantly (see section about experiments).

The following equivalence classes can be used for constraints when using untagged corpora:

1. States that generate the same outputs (unigram constraint)
2. unigram constraint, and additionally all predecessor states must generate the same outputs (bigram constraint)
3. trigrams or higher, if the corpora are large enough
4. a variation of one: states that output words belonging to one ambiguity class, i.e. can be of a certain number of syntactic classes.

Merging starts with one of the constraints. After a number of merges have been performed, the constraint is discarded and a weaker one is used instead.

The standard $n$-gram approaches are special cases of using model merging and constraints. E.g., if we use the unigram constraint, and merge states until no further merge is possible under this constraint, the resulting model is a standard bigram model, regardless of the order in which the merges were performed.

In practice, a constraint will be discarded before no further merge is possible (otherwise the model could have been derived directly, e.g., by the standard $n$-gram technique). Yet, the question when to discard a constraint to achieve best results is unsolved.

### The Starting Point

The initial model of the original model merging procedure is the maximum likelihood or trivial model. This model has the advantage of directly representing the corpus. But its disadvantage is its huge number of states. A lot of computation time can be saved by choosing an initial model with fewer states.

The initial model must have two properties:

1. it must be larger than the intended model, and
2. it must be easy to construct.

The trivial model has both properties. A class of models that can serve as the initial model as well are $n$-gram models. These models are smaller by one or more orders of magnitude than the trivial model and therefore could speed up the derivation of a model significantly.

This choice of a starting point excludes a lot of solutions which are allowed when starting with the maximum likelihood model. Therefore, starting with an $n$-gram model yields a model that is at most equivalent to one that is generated when starting with the trivial model, and that can be much worse. But it should be still better than any $n$-gram model that is of lower of equal order than the initial model.

## Experiments

### Model Merging vs. Bigrams

The first experiment compares model merging with a standard bigram model. Both are trained on the same data. We use $N_{train} = 14,421$ words of the Verbmobil corpus. The corpus consists of transliterated dialogues on business appointments[1]. The models are tested on $N_{test} = 2,436$ words of the same corpus. Training and test parts are disjunct.

The bigram model yields a Markov model with 1,440 states. It assigns a log perplexity of 1.20 to the training part and 2.40 to the test part.

Model merging starts with the maximum likelihood model for the training part. It has 14,423 states, which correspond to the 14,421 words (plus an initial and a final state). The initial log perplexity of the training part is 0.12. This low value shows that the initial model is very specialized in the training part.

[1] Many thanks to the Verbmobil project for providing these data. We use dialogues that were recorded in 1993 and 94, and which are now available from the Bavarian Archive for Speech Signals BAS (http://www.phonetik.uni-muenchen.de/Bas/BasHomeeng.html).

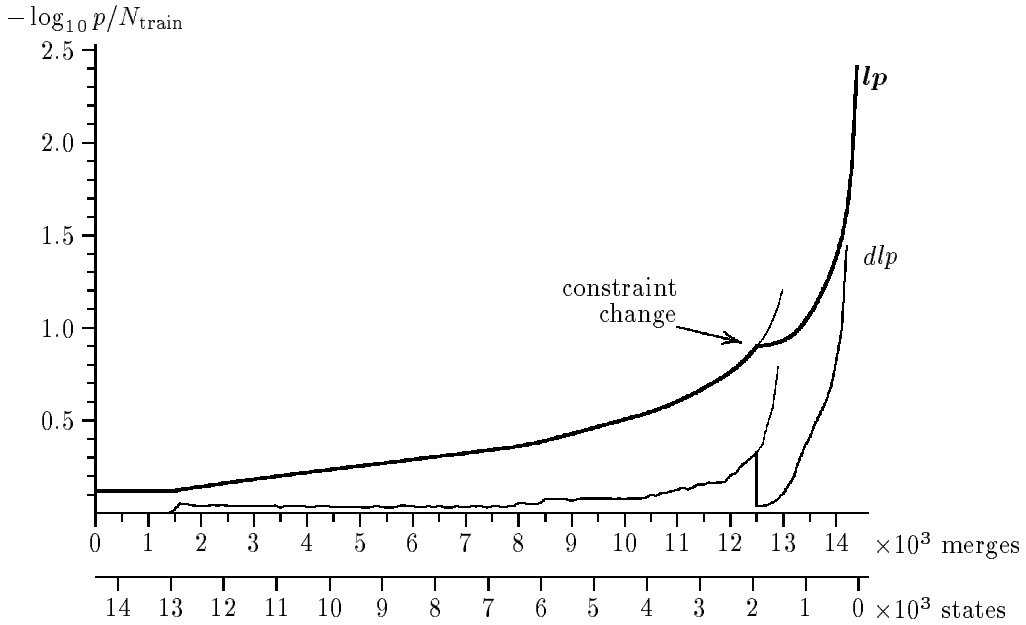


Figure 2: Log Perplexity of the training part during merging. Constraints: same output until 12,500 / none after 12,500. The thin lines show the further development if we retain the the same-output constraint until no further merge is possible. The length of the training part is $N_{\text{train}} = 14,421$.

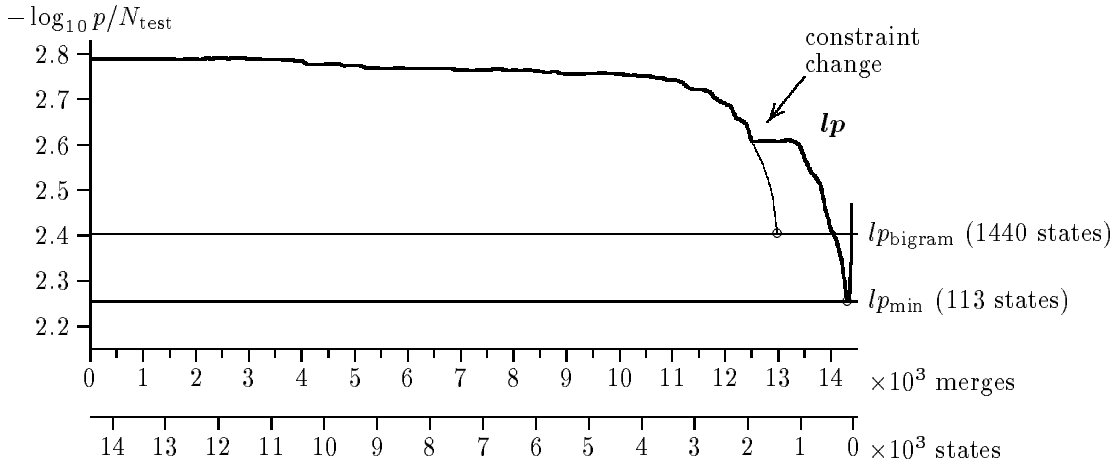


Figure 3: Log Perplexity of Test Part During Merging. Constraints: Same Output until 12,500 / none after 12,500. The thin line shows the further development if we retain the same-output constraint, finally yielding a bigram model. The length of the test part is $N_{\text{test}} = 2,436$.

We start merging with the same-output (unigram) constraint to reduce computation time. After 12,500 merges the constraint is discarded and from then on all remaining states are allowed to merge. The constraints and the point of changing the constraint are chosen for pragmatic reasons. We want the constraints to be as week as possible to allow the maximal number of solutions but at the same time the number of merges must be manageable by the system used for computation (a SparcServer1000 with 250MB main memory). As the following experiment will show, the exact points of introducing/discarding constraints is not important for the resulting model.

There are $N_{train}(N_{train}-1)/2 \sim 10^8$ hypothetical first merges in the unconstraint case. This number is reduced to $\sim 7 \cdot 10^5$ when using the unigram constraint, thus by a factor of $\sim 150$. By using the constraint we need about a week of computation time on a SparcServer 1000 for the whole merging process. Computation would not have been feasible without this reduction.

Figure 2 shows the increase in perplexity during merging. There is no change during the first 1,454 merges. Here, only identical sequences of initial and final states are merged (compare figure 1.a to c). These merges do not influence the probability assigned to the training part and thus do not change the perplexity.

Then, perplexity slowly increases. It can never decrease: the maximum likelihood model assigns the highest probability to the training part and thus the lowest perplexity.

Figure 2 also shows the perplexity's slope. It is low until about 12,000 merges, then drastically increases. At about this point, after 12,500 merges, we discard the constraint. For this reason, the curve is discontinuous at 12,500 merges. The effect of further retaining the constraint is shown by the thin lines. These stop after 12,983 merges, when all states with the same outputs are merged (i.e., when a bigram model is reached). Merging without a constraint continues until only three states remain: the initial and the final state plus one proper state.

Note that the perplexity changes very slowly for the largest part, and then changes drastically during the last merges. There is a constant phase between 0 and 1,454 merges. Between 1,454 and ~11,000 merges the log perplexity roughly linearly increases with the number of merges, and it explodes afterwards.

What happens to the test part? Model merging starts with a very special model which then is generalized. Therefore, the perplexity of some random sample of dialogue data (what the test part is supposed to be) should decrease during merging.

Table 1: Number of states and Log Perplexity for the derived models and an additional, previously test part, consisting of 9,784 words. (a) standard bigram model, (b) constrained model merging (first experiment), (c) model merging starting with a bigram model(second experiment)

| | (a) | (b) | (c) |
|---|---|---|---|
| type | bigrams | model merging | MM start with bigrams |
| # states | 1,440 | 113 | 113 |
| Log PP | 2.78 | 2.41 | 2.39 |

This is exactly what we find in the experiment.

Figure 3 shows the log perplexity of the test part during merging. Again, we find the discontinuity at the point where the constraint is changed. And again, we find very little change in perplexity during about 12,000 initial merges, and large changes during the last merges.

Model merging finds a model with 113 states, which assigns a log perplexity of 2.26 to the test part. Thus, in addition to finding a model with lower log perplexity than the bigram model (2.26 vs. 2.40), we find a model that at the same time has less than 1/10 of the states (113 vs. 1,440).

To test if we found a model that predicts new data better than the bigram model and to be sure that we did not find a model that is simply very specialized to the test part, we use a new, previously unseen part of the Verbmobil corpus. This part consists of 9,784 words. The bigram model assigns a log perplexity of 2.78, the merged model with 113 states assigns a log perplexity of 2.41 (see table 1). Thus, the model found by model merging can be regarded generally better than the bigram model.

## Improvements

The derivation of the optimal model took about a week although the size of the training part was relatively small. Standard speech applications do not use 14,000 words for training as we do in this experiment, but 100,000, 200,000 or more. It is not possible to start with a model of 100,000 states and to successively merge them, at least it is not possible on today's machines. Each step would require the test of $\approx 10^9$ merges.

In the previous experiment, we abandoned the same-output constraint after 12,500 merges to keep the influence on the final result as small as possible. It can not be skipped from the beginning because somehow the time complexity has to be reduced. But it can be further retained, until no further merge under this constraint is possible.

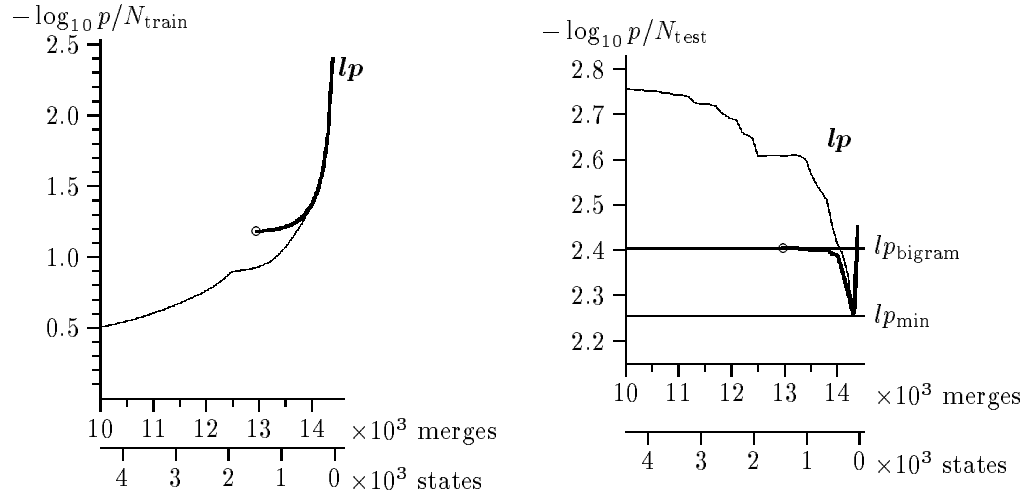


Figure 4: Log Perplexity of training and test parts when starting with a bigram model. The starting point is indicated with $\circ$, the curves of the previous experiment are shown in thin lines.

This yields a bigram model. The second experiment uses the bigram model with 1,440 states as its starting point and imposes no constraints on the merges. The results are shown in figure 4.

We see that the perplexity curves approach very fast their counterparts from the previous experiment. The states differ from those of the previously found model, but there is no difference in the number of states and corpus perplexity in the optimal point. So, one could in fact, at least in the shown case, start with the bigram model without loosing anything. Finally, we calculate the perplexity for the additional test part. It is 2.39, thus again lower than the perplexity of the bigram model (see table 1). It is even slightly lower than in the previous experiment, but most probably due to random variation.

The derived models are not in any case equivalent (with respect to perplexity), regardless whether we start with the trivial model or the bigram model. We ascribe the equivalence in the experiment to the particular size of the training corpus. For a larger training corpus, the optimal model should be closer in size to the bigram model, or even larger than a bigram model. In such a case starting with bigrams does not lead to an optimal model, and a trigram model must be used.

## Conclusion

We investigated model merging, a technique to induce Markov models from corpora. The original procedure is improved by introducing constraints and a different initial model. The procedures are shown to be applicable to a transliterated speech corpus. The derived models assign lower perplexities to test data than the standard bigram model derived from the same training corpus. Additionally, the merged model was much smaller than the bigram model.

The experiments revealed a feature of model merging that allows for improvement of the method's time complexity. There is a large initial part of merges that do not change the model's perplexity w.r.t. the test part, and that do not influence the final optimal model. The time needed to derive a model is drastically reduced by abbreviating these initial merges. Instead of starting with the trivial model, one can start with a smaller, easy-to-produce model, but one has to ensure that its size is still larger than the optimal model.

## Acknowledgements

I would like to thank Christer Samuelsson for very useful comments on this paper. This work was supported by the Graduiertenkolleg Kognitionswissenschaft, Saarbrücken.

## References

[Bahl et al., 1983] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190.

[Baum et al., 1970] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occuring in the statistical analysis of probabilistic functions in markov

chains. *The Annals of Methematical Statistics*, 41:164–171.

[Brants, 1995] Thorsten Brants. 1995. Estimating HMM topologies. In *Tbilisi Symposium on Language, Logic, and Computation*, Human Communication Research Centre, Edinburgh, HCRC/RP-72.

[Church, 1988] Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, Texas, USA.

[Cutting et al., 1992] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ACL)*, pages 133–140.

[Jelinek, 1990] F. Jelinek. 1990. Self-organized language modeling for speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Kaufmann, San Mateo, CA.

[Omohundro, 1992] S. M. Omohundro. 1992. Best-first model merging for dynamic learning and recognition. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 958–965. Kaufmann, San Mateo, CA.

[Pereira et al., 1993] Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of the 31st ACL*, Columbus, Ohio.

[Rabiner, 1989] L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–285.

[Stolcke and Omohundro, 1994] Andreas Stolcke and Stephen M. Omohundro. 1994. Best-first model merging for hidden markov model induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley, California, USA.

[Viterbi, 1967] A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *IEEE Transactions on Information Theory*, pages 260–269.